Robust Adaptive Kinematic Control of Redundant Robots *

M. Tarokh and D. D. Zuck
Robotics and Intelligent Systems Laboratory
Department of Mathematical Sciences
San Diego State University
San Diego, CA 92182

Abstract

The paper presents a general method for the resolution of redundancy that combines the Jacobian pseudoinverse and augmentation approaches. A direct adaptive control scheme is developed to generate joint angle trajectories for achieving desired end-effector motion as well as additional user defined tasks. The scheme ensures arbitrarily small errors between the desired and the actual motion of the manipulator. Explicit bounds on the errors are established that are directly related to the mismatch between actual and estimated pseudoinverse Jacobian matrix, motion velocity and the controller gain. It is shown that the scheme is tolerant of the mismatch and consequently only infrequent pseudoinverse computations are needed during a typical robot motion. As a result, the scheme is computationally fast, and can be implemented for real-time control of redundant robots. A method is incorporated to cope with the robot singularities allowing the manipulator to get very close or even pass through a singularity while maintaining a good tracking performance and acceptable joint velocities. Computer simulations and experimental results are provided in support of the theoretical developments.

1 Introduction

The dexterity and versatility offered by redundant manipulators allow their utilization for the performance of complex tasks in practical environments. However, effective utilization of this dexterity requires satisfactory resolution of the redundancy and its real-time implementation.

During recent years two main approaches to the resolution of the redundancy have emerged. These can be categorized as Jacobian pseudoinverse [1]-[9] and Jacobian augmentation [10]-[14] approaches. In the pseudoinverse approach, a certain vector lying in the null space of the Jacobian matrix is utilized for a variety of design objectives. These objectives include optimization of a performance criterion [2], obstacle avoidance [3], torque optimization [4], and task prioritization [5]-[6]. A review of pseudoinverse methods is given in [7]. In the augmented Jacobian approach, an additional Jacobian matrix is defined for the purpose of utilizing the extra degrees of freedom offered by redundancy. This matrix is augmented with the end-effector Jacobian matrix to obtain a square Jacobian matrix, and thus the problem of redundant manipulator control is transformed to that of a non-redundant manipulator. A method to augment the Jacobian matrix for the purpose

*This work was supported by NETROLOGIC Inc through NASA Grant No. NAS7-1110

of optimizing a performance criterion is proposed by Baileiul [11]-[12]. The concept of augmenting the Jacobian matrix is generalized by Seraji [13], allowing the utilization of the redundancy for achieving a variety of objectives [14]. The augmented Jacobian approach has the feature of making the motion cyclic, which is desirable for repetitive operation, and presents an advantage over the pseudoinverse approach. However, augmentation introduces additional singularities which cannot be easily characterized and which aggravate the singularity problem associated with revolute joint manipulators. Desired Cartesian trajectories passing in the neighborhood of such a singularity demand very large joint velocities which are impossible to achieve in practice. To overcome the singularity problem, methods have been proposed [6], [15] that reduce the joint velocities at the cost of introducing or increasing the mismatch between the computed and the actual inverse Jacobian matrix. Such a mismatch produces errors in position and orientation when attempting to control the manipulator.

Motion control of a redundant manipulator can be implemented in a hierarchical scheme using either of the above two approaches to the redundancy resolution. In such a scheme the joint angle trajectories are generated to achieve a desired end-effector motion, as well as achieving additional objectives offered by extra degrees of freedom. The generated joint angles are then used as the set points of the low level servo-loops. Such a hierarchical scheme is particularly attractive in practice since most industrial manipulators have high performance servo-loops that readily accept joint angle set points but cannot easily be modified to implement joint torques in a non-hierarchical scheme. A joint space trajectory generator using the feedback control approach was originally proposed in [16], and extended to redundant robots in [17]-[19] within the framework of the pseudoinverse approach and in [20] using the augmented Jacobian approach.

Regardless of the approach used to resolve the redundancy and to overcome the singularity problem, the computations involved in motion control of a redundant manipulator can be excessive. Motion control using the pseudoinverse approach requires computation of the Jacobian pseudoinverse, its null space matrix, and derivative of an objective function at every control cycle. Similarly, motion control using the augmented Jacobian approach, requires determining the Jacobian matrix associated with user defined kinematic functions, and the inverse of a higher dimensional augmented Jacobian matrix at each control cycle. These intensive computations can make real time implementation of motion control on a practical redundant manipulator impossible.

In this paper we propose a general approach to the redundancy resolution which retains the essential features of both the pseudoinverse and the augmentation methods, and which reduces to either method as a special case. Within the framework of this general approach, an adaptive kinematic control scheme is developed for trajectory tracking that requires only a crude estimate of the inverse Jacobian matrix, and thus allows very infrequent computation of the inverse or the pseudoinverse matrix. This results in considerable computational savings and makes real-time implementation of the scheme feasible on a practical redundant robot. The kinematic control scheme also achieves high tracking accuracies and acceptable joint velocities even when the manipulator passes through

a singularity.

2 Adaptive Kinematic Control

Consider an n jointed robot manipulator performing tasks in the operational space. The relationship between the $m_e \times 1$ end-effector position and orientation vector X_e , and the $n \times 1$ joint space vector Θ, where $m_e \leq n$, is given by the forward kinematic map

$$X_e = f_e(\Theta) \tag{1}$$

The corresponding relationship for velocities is

$$\dot{X}_e = J_e(\Theta)\dot{\Theta} \tag{2}$$

where $J_e(\Theta) = \frac{\partial f_e}{\partial \Theta}$ is the $m_e \times n$ Jacobian matrix of the end-effector. The problem of kinematic control of a redundant robot is to determine the joint angle vector $\Theta(t)$ to achieve a desired end-effector trajectory vector $X_{ed}(t)$, and to utilize the redundancy offered by $r = n - m_e$ extra degrees of freedom to perform additional tasks. In the pseudoinverse method of redundancy resolution, the joint velocity vector $\dot{\Theta}(t)$ is related to the end-effector velocity $\dot{X}_e$ by

$$\dot{\Theta}(t) = G_e(\Theta)\dot{X}_e(t) + \gamma(I_n - G_e(\Theta)J_e(\Theta))Z \tag{3}$$

where $G_e(\Theta) \equiv J_e^{\dagger}(\Theta)$ is the pseudoinverse of $J_e(\Theta)$ satisfying $J_eG_eJ_e = J_e$, $J_eG_e = G_e$, $(G_eJ_e)^{\dagger} = J_eG_e$ and $(J_eG_e)^{\dagger} = G_eJ_e$, and the argument Θ has been dropped for convenience. The scalar γ is a positive weighting factor, and Z is an arbitrary $n \times 1$ vector that has no effect on the end-effector motion due to the fact that it is multiplied by the null space of J_e. In the pseudoinverse method, this vector is generally set to the gradient of an objective function $\Psi(\Theta)$ for the purpose of optimization, that is $Z = \frac{\partial \Psi}{\partial \Theta}$. In the generalized augmentation method proposed by Seraji [13], r additional kinematic functions $X_a = f_a(\Theta)$ are defined to resolve the redundancy. These functions are chosen to reflect the desired additional tasks to be performed. The $r \times n$ Jacobian matrix of additional tasks $J_a(\Theta) = \frac{\partial f_a}{\partial \Theta}$ is augmented with the $m_e \times n$ end-effector Jacobian matrix J_e to form an augmented $n \times n$ Jacobian matrix. The manipulator now becomes non-redundant. The generalized augmentation method has the advantage of letting the user easily define additional kinematic functions and making the motion cyclic. However, the utilization of redundancy for optimization, although theoretically possible within the framework of the augmented Jacobian method, is computationally very intensive. This is due to the fact that the Jacobian matrix must be augmented with the matrix $J_a = \frac{\partial}{\partial \Theta}\left(N^T \frac{\partial \Psi}{\partial \Theta}\right)$, where $N = (I_n - G_eJ_e)$ is the null space of the end-effector Jacobian matrix.

In general, a redundant manipulator can be utilized for achieving two types of tasks. The first type, which we will refer to as the primary tasks, are those tasks that can be expressed by a set of kinematic equality constraints that must be satisfied accurately. Examples of such tasks are tracking an end-effector trajectory, and maintaining an elbow height or a shoulder angle at specified values for

the achievement of a certain objective. The other type of tasks, or secondary tasks, involve realizing a performance criterion as best as possible, for example, optimizing a performance criterion to avoid joint limits. These tasks do not need to be accurately monitored or tightly controlled, and an approximate optimization is generally acceptable. In the augmentation method all additional tasks are treated as primary tasks that must to be achieved accurately. An excessive number of primary tasks will result in the difficulty or inability in achieving these tasks. On the other hand, in the pseudoinverse method, the tasks are formulated in the form of an optimization criterion, which may not be the most practical or natural way of expressing and solving a particular kinematic problem.

To ease these trade-offs, we propose a method for combining augmentation and pseudoinverse approaches, thereby permitting a more natural formulation of both kinds of tasks. Specifically, let $0 \leq r_a \leq r$ degrees of redundancy be used for the user defined tasks described by the $r_a \times 1$ vector equation $X_a = f_a(\Theta)$, where X_a is the additional task vector. The augmented system of kinematic equations is

$$X = \begin{pmatrix} X_e \\ X_a \end{pmatrix} = \begin{pmatrix} f_e(\Theta) \\ f_a(\Theta) \end{pmatrix} \tag{4}$$

where X is an $m \times 1$ vector, $m = (m_e + r_a)$, and will be referred to as the "posture" vector. This vector is composed of position and orientation of the end-effector, and possible additional kinematic functions that define Cartesian or angular position of the arm such as elbow height or distance of points on the arm from obstacles. The equation relating $\dot{X}$ to $\dot{\Theta}$ is

$$\dot{X} = J\dot{\Theta} \tag{5}$$

where $J = \begin{pmatrix} J_e \\ J_a \end{pmatrix}$ is the $m \times n$ augmented Jacobian matrix. Similarly, let $0 \leq r_o \leq r$ degrees of freedom be utilized for optimization of a performance criterion $\Psi(\Theta)$, where $r = r_a + r_o$. The optimization criterion can be manipulability maximization, joint limit avoidance, minimal joint motion or obstacle avoidance, to mention a few. Equation (3) can now be used to obtain the joint angle trajectory vector $\Theta(t)$ for achieving both a desired posture vector $X_d(t)$ and optimization of the performance criteria $\Psi(\Theta)$. This gives

$$\Theta(t) = \int_0^t \left(G\dot{X}_d(t) + \gamma(I_n - GJ)Z \right) dt \tag{6}$$

where $G \equiv J^{\dagger}(\Theta)$ and $Z = \frac{\partial \Psi}{\partial \Theta}$. It is evident that when $r_o = 0$, the method reduces to the purely augmented Jacobian matrix method. Likewise, when $r_a = 0$, it reduces to purely pseudoinverse method.

Equation (6) is of little practical use since the integration can drift away even with small inaccuracies in the knowledge of the kinematic parameters or the computation of the Jacobian matrix. It will also produce undesirable behavior when the manipulator passes close to a singularity. It is, therefore, important to develop a scheme that is robust with respect to robot singularities or inaccuracies in the estimation or computation of the Jacobian matrix. The latter is very important

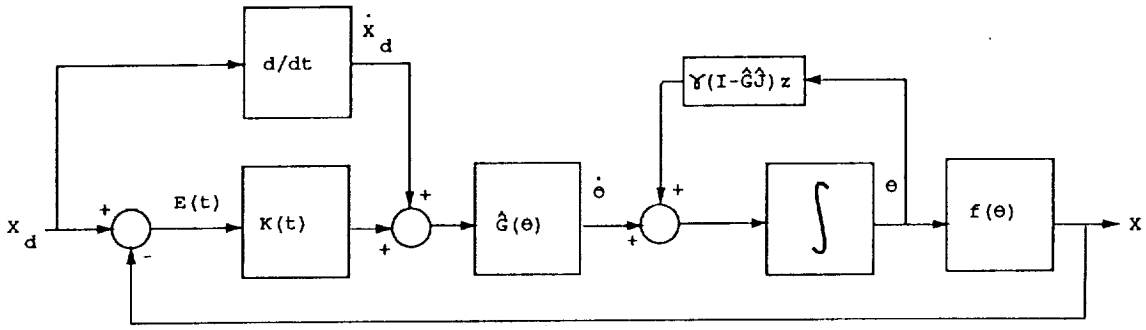

Figure 1: Block diagram of the overall scheme.

since the computation of the Jacobian inverse or the pseudoinverse must be performed very infrequently to reduce computational burden. Such infrequent computations generate inaccuracies in the inverse Jacobian matrix and are reflected as errors in the manipulator position and orientation. In order to resolve these difficulties, the feedback control scheme of Figure 1 is employed, where $E(t) = X_d(t) - X(t)$ denotes the error between the desired and actual posture vector. The controller consists of an adaptive time-varying feedback gain matrix $K(t)$ acting on the error and a matrix $\hat{G}$ that represents an estimate of the pseudoinverse Jacobian matrix $G = J^{\dagger}$. The estimate $\hat{G}$ can be considerably different from the actual G, although a good estimate reduces the controller gain, as will be seen. As we discussed earlier, our objective is to achieve accurate tracking of the end-effector and the additional tasks as defined by the desired value of the posture vector $X_d(t)$, while attempting to satisfy the secondary tasks as best as possible. Consequently, the error in the posture vector is directly controlled, whereas the optimization is indirectly realized. Figure 1 indicates that the required joint angular velocity vector $\dot{\Theta}(t)$ can be obtained from

$$\dot{\Theta}(t) = \hat{G}\left(\dot{X}_d(t) + K(t)E(t)\right) + \gamma(I_n - \hat{G}\hat{J})Z \tag{7}$$

where $\hat{J}$ is an estimate of the Jacobian matrix. The problem is now to determine $K(t)$ to ensure that the posture vector $X(t)$ closely follows its desired value $X_d(t)$ so that the error $E(t)$ is arbitrarily small. Premultiplying (7) by J and substituting the result into (5), we obtain

$$\dot{X}(t) = J\hat{G}\left[\dot{X}_d(t) + K(t)E(t)\right] + \gamma J(I_n - \hat{G}\hat{J})Z \tag{8}$$

Subtracting both sides of (8) from $\dot{X}_d$ and rearranging yields the error dynamic equation

$$\dot{E}(t) = -J\hat{G}K(t)E(t) + (I_m - J\hat{G})\dot{X}_d(t) - \gamma J(I_n - \hat{G}\hat{J})Z \tag{9}$$

Let us define the mismatch between the actual and estimated Jacobian matrices as

$$H = I_m - J\hat{G} \tag{10}$$

In the ideal case where J is known accurately and $rank\ J = m$, the mismatch matrix is $H = I_m - JJ^T(JJ^T)^{-1} = 0$, and there is no mismatch. Substituting (10) into (9) and simplifying, we obtain

$$\dot{E}(t) = -K(t)E(t) + HK(t)E(t) + H\dot{X}_d(t) - \gamma H_1 z \tag{11}$$

where $H_1 = (J - \hat{J}) + H\hat{J}$ is a modified mismatch matrix and has the property that $H_1 = 0$ whenever the original mismatch matrix H is zero. The term $\gamma H_1 z$ is in fact the interaction of the secondary task on the primary task error dynamics. When $H = 0$, (11) simplifies to $\dot{E}(t) = -K(t)\,E(t)$. In this case the nonlinear system (11) is reduced to a simple linear system, which partially explains the reason for using the feedback configuration of Figure 1. It is easy to show that in the case of zero mismatch, $lim_{t\to\infty} E = 0$, provided that $K(t)$ is a symmetric positive definite matrix. However, there will always be a mismatch because of the imperfect knowledge of the robot parameters, the inaccuracies due to infrequent calculations of the inverse Jacobian matrix, or the robot operating near a singular point. It is, therefore, desirable to develop a posture trajectory tracking algorithm that will be robust to both inaccuracies in the Jacobian matrix calculations and robot singularities. In the next section, we will show that the control algorithm (7) with $K(t)$ designed as an adaptive proportional plus integral controller will achieve these objectives.

3 Stability and Tracking Performance

Consider the proportional plus integral feedback matrix

$$K(t) = K_P + K_I(t) \tag{12}$$

where K_P is a constant positive definite symmetric matrix, and $K_I(t)$ is obtained from an integral adaptation algorithm as

$$K_I(t) = K_0 + \int_0^t \left(\alpha E(\tau)E^T(\tau) - \sigma K_I(\tau)\right) d\tau \tag{13}$$

where K_0 is a constant positive definite symmetric matrix representing the initial value of the integral, $\alpha > 0$ is the constant integral coefficient, and $\sigma > 0$ is the leakage coefficient used to avoid possible integral wind up [21]. Equation (13) implies that

$$\dot{K}_I(t) = \alpha\, E(t)E^T(t) - \sigma K_I(t) \tag{14}$$

Note that $K_I(t)$ is a positive definite symmetric matrix for all t. In order to show that the error $E(t)$ described by the dynamical equations (11) and (14) can be made arbitrarily small, we consider the Lyapunov function candidate

$$V = \frac{1}{2}E^T(t)E(t) + \frac{1}{2\alpha^2}\, tr\left(K_I^T(t)K_I(t)\right) \tag{15}$$

whose derivative along the trajectories of (11) and (14) is

$$\dot{V} = -E^T KE + E^T HKE + E^T H\dot{X}_d - \gamma E^T H_1 Z + \frac{1}{\alpha^2}\, tr\left(K_I^T \dot{K}_I\right) \tag{16}$$

where the time argument is dropped for convenience. Substituting for K and $\dot{K}$ from (12) and (14) into (16) and simplifying, we obtain

$$\dot{V} = -E^T \left((1 - \frac{1}{\alpha}) K_I + K_P \right) E + E^T H K E + E^T H \dot{X}_d - \gamma E^T H_1 Z - \frac{\sigma}{\alpha^2} tr \left(K_I^T K_I \right) \tag{17}$$

Suppose that the coefficient α is chosen such that $\alpha > 1$, and let $k_1 = \lambda_{min} \left(K_P + (1 - \frac{1}{\alpha}) K_I \right)$, and $k_2 = \lambda_{max}(K)$, where λ_{min} and λ_{max} denote the minimum and the maximum eigenvalues of a matrix, respectively. Equation (17) implies that

$$\dot{V} \leq -k_1 \|E\|^2 + k_2 \|H\| \|E\|^2 + E^T \left(H \dot{X}_d - \gamma H_1 Z \right) - \frac{\sigma}{\alpha^2} tr \left(K_I^T K_I \right) \tag{18}$$

Let $\eta = max(\|H\|)$, $\eta_1 = max(\|H_1\|)$, $\zeta = max(\|\dot{X}_d\|)$, and $\rho = max(\|Z\|)$. We have in the worst case

$$\dot{V} \leq -k_1 \left(1 - \frac{k_2}{k_1} \eta \right) \|E\|^2 + (\eta \zeta + \gamma \eta_1 \rho) \|E\| - \frac{\sigma}{\alpha^2} tr \left(K_I^T K_I \right) \tag{19}$$

In order to prove the stability of the error dynamics, we will assume that $\eta < \frac{k_1}{k_2} \leq 1$. This assumption places an upper bound on the mismatch. It must be noted, however, that this upper bound is resulted from the worst case nature of the analysis, and that in practice the scheme often accommodates larger mismatches, as will be demonstrated in Section 4. Let

$$k = k_1 \left(1 - \frac{k_2}{k_1} \eta \right) \quad ; \quad q = \eta \zeta + \gamma \eta_1 \rho \tag{20}$$

to obtain from (19)

$$\dot{V} \leq -k \|E\|^2 + q \|E\| - \frac{\sigma}{\alpha^2} tr \left[K_I^T K_I \right) \tag{21}$$

Substituting for $\|E\|^2 = 2V - \frac{1}{\alpha^2} tr \left(K_I^T K_I \right)$ from (15) into (21) and dropping the negative term $-\frac{1}{\alpha^2} tr \left(K_I^T K_I \right)$ appearing under the radical in the resulting equation, we obtain

$$\dot{V} \leq -2kV + q \sqrt{2V} - \left(\frac{\sigma - k}{\alpha^2} \right) tr \left(K_I^T K_I \right) \tag{22}$$

Now if the leakage coefficient is selected such that $\sigma \geq k$, then (22) reduces to

$$\dot{V} \leq -2kV + q \sqrt{2V} \tag{23}$$

Note that the choice of $\sigma \geq k$ is very conservative due to the fact that the negative term was dropped above and that in practice a much smaller value of σ is sufficient to ensure the validity of inequality (23). This inequality is now in a form that can be solved analytically by defining

$$W = \sqrt{V} - \frac{q}{\sqrt{2k}} \tag{24}$$

which yields

$$\dot{W} = \frac{\dot{V}}{2\sqrt{V}} \tag{25}$$

Substituting (24) and (25) into (23) and dividing both sides of the resulting inequality by $2\sqrt{V} > 0$, we obtain

$$\dot{W} \leq -kW \tag{26}$$

The solution to (26) is

$$W(E) \leq W(E(t_0))\, e^{-k(t-t_0)} \tag{27}$$

where t_0 is the initial time. In view of (24), the last inequality becomes

$$V(E) \leq \left[e^{-k(t-t_0)} \sqrt{V(E(t_0))} + \left(1 - e^{-k(t-t_0)}\right) \frac{q}{\sqrt{2}k} \right]^2 \tag{28}$$

It is easy to verify that $V(E)$ decreases monotonically along any solution of the error dynamic equations (11) and (14) until the solution reaches the compact region

$$\mathcal{R}_f = \left\{ E : V(E) \leq V_f = \frac{q^2}{2k^2} \right\} \tag{29}$$

where the subscript f denotes the final values, that is as $t \to \infty$. The rate of decrease of $V(E)$ is at least as fast as e^{-kt}. We conclude that the solutions E of (11) and (14) are globally ultimately bounded . Since from (15) we have $\|E\|^2 \leq 2V$, it follows that within the region $\mathcal{R}_f$ given by (29), the error is bounded by

$$\|E\| \leq \frac{q}{k} \tag{30}$$

Equations (30) and (20) indicate that the upper bound on the steady-state tacking error is related to the accuracy of the inverse Jacobian matrix estimation, the desired velocity of the posture vector, and the controller gain. The adaptive nature of K coupled with a reasonable estimate of the inverse Jacobian allows the achievement of very low tracking error with small or moderate controller gains. Extensive simulation results have shown that even when the estimate of the Jacobian matrix is significantly different from its actual value and the end-effector moves fast, the error can be made very small by increasing the controller gain k. The latter can be achieved using higher values of the integral coefficient α and the constant matrices K_0 and K_P.

3.1 Discrete-Time Implementation

In practice, the adaptive trajectory generator algorithm described by (7), (12) and (13) is implemented in discrete time. Using the trapezoidal integration rule, the discrete form of (13) is

$$K_{Ii} = K_0 + \frac{T_c}{2} \sum_{j=0}^{i-1} \left[\alpha \left(E_j E_j^T + E_{j+1} E_{j+1}^T \right) - \sigma \left(K_{Ij} + K_{I(j+1)} \right) \right] \tag{31}$$

where T_c is the control sample time and K_{Ii} and E_i denote , respectively, the values of the adaptive matrix $K_I(t)$ and the error vector $E(t)$ at time $t = iT_c$, $i = 0, 1, 2, \ldots$. As with any discrete time feedback systems, the tracking performance deteriorates when T_c is increased. Equation (31) can be written in a computationally more convenient form as

$$K_{Ii} = \left(\frac{1}{1 + 0.5\sigma T_c} \right) \left[(1 - 0.5\sigma T_c) K_{I(i-1)} + 0.5\alpha T_c (E_i E_i^T + E_{i-1} E_{i-1}^T) \right] \tag{32}$$

Similarly, the discrete form of the control law (7) using the trapezoidal integration rule is

$$\Theta_i = \Theta_{i-1} + \frac{T_c}{2}(\Omega_i + \Omega_{i-1}) \tag{33}$$

where

$$\Omega_i = \hat{G}_i \left(\dot{X}_{di} + K_i E_i \right) + \gamma (I_n - \hat{G}_i \hat{J}_i) Z_i \tag{34}$$

where the subscript i denotes the values at time $t = iT_c$. The above control law requires an estimate of the Jacobian pseudoinverse matrix $\hat{G}_i$. An efficient algorithm to calculate the Jacobian matrix $\hat{J}_i$ is given in [22]. The estimated pseudoinverse matrix is obtained from J_i as

$$\hat{G}_i = \hat{J}_i^T \left(\hat{J}_i \hat{J}_i^T \right)^{-1} \tag{35}$$

where the pseudoinvere in (35) minimizes $\| \dot{X}_i - \hat{J}_i \dot{\Theta}_i \|$. This procedure will provide a reasonably accurate estimate of the pseudoinverse, that is $\hat{G} \approx G$. However, it has two main drawbacks, namely computational intensity and excessively large values of $\hat{G}$, and consequently of joint velocities, near the robot singularities. In order to resolve the problem of excessive joint velocities near the robot singularities, we estimate the Jacobian pseudoinverse to minimize

$$\{ \| \dot{X}_i - J_i \dot{\Theta}_i \| + \beta \| \dot{\Theta}_i \| \} \tag{36}$$

where the value of $\beta \geq 0$ determines the weighting placed on the minimization of joint velocity errors. The inverse Jacobian matrix that realizes (36) is [6]

$$\hat{G}_i = J_i^T \left(J_i J_i^T + \beta I \right)^{-1} \tag{37}$$

Note that the inverse in (37) exists even when the manipulator is at a singularity. The weighting factor β can be adjusted to have a small value near robot singularity, and zero elsewhere. One method to achieve this is to choose β according to

$$\beta = \begin{cases} \beta_0 (1 - \frac{w}{w_0})^2, & w < w_0 \\ 0, & w \geq w_0 \end{cases} \tag{38}$$

where $w = \sqrt{det(JJ^T)}$ is the manipulability measure, β_0 is a constant and w_0 is a specified threshold. A more elaborate technique for the selection of β is based on singular values of J_i [3]. The addition of the term βI in (37) introduces mismatch in the Jacobian pseudoinverse and can produce tracking errors. However, the proposed adaptive kinematic scheme is robust with respect to the mismatch and will automatically increase the gains to the level that is needed to achieve good tracking performance, as discussed before.

A major portion of the computation in each control cycle is the calculation of the Jacobian matrix and its pseudoinverse. In order to significantly reduce the computational burden, the calculation of the Jacobian matrix and its pseudoinverse are performed, merely once every $T_J = \nu T_c$ seconds, where $\nu \gg 1$ (typically 100) is an integer. This implies that the costly computation of the Jacobian matrix

and its pseudoinverse are performed only a few times during a typical robot motion. Furthermore, to account for the delay time that occurs due to the computation of $\hat{G}_i$, we introduce a νT_c second delay in the implementation of the scheme. Equation (34) is now

$$\Omega_i = \hat{G}_{l-1}\left(\dot{X}_{di} + K_i E_i\right) + \gamma(I_n - \hat{G}_{l-1}\hat{J}_{l-1})Z_i \tag{39}$$

where $\hat{G}_{l-1}$ is the value of $\hat{G}$ at time $t = (l-1)T_j = (l-1)\nu T_c$, $l = 1, 2, 3, \cdots$. Because of the robustness of the adaptive kinematic algorithm to the Jacobian mismatch, accurate tracking is possible despite the mismatch which is brought about by both the infrequent updating of Jacobian pseudoinverse and the computation delays. This will be demonstrated in the next section.

4 Examples

In this section, the adaptive joint space trajectory generator developed in the previous sections will be applied to two examples. In the first example a modified PUMA 562 is utilized to demonstrate the redundancy resolution using the proposed adaptive kinematic control method. In the second example, the kinematic model of a regular PUMA 562 manipulator is used for both position and orientation tracking while the robot passes repeatedly through singularities. The control sample time is selected as $T_c = 2\,ms$ in all cases.

4.1 Example 1

In this example we investigate the performance of the adaptive scheme for trajectory tracking of a modified PUMA 562 manipulator. The PUMA is made redundant by adding an extra link to its wrist, as shown in Figure 2. The joint angles to be used are waist θ_1, shoulder θ_2, elbow θ_3, and wrist angles θ_4 and θ_5. Joint 6 provides rotation of the extra link and will not be used in this example. The modified PUMA is now a redundant for positioning the tip of the extra link in the three dimensional space. The kinematic equations of the modified PUMA, which has two degrees of redundancy, are given in the Appendix. In the following we will study two cases, where in both cases the tip is required to move on a straight line in the Cartesian space. The desired position trajectory is described by

$$X_{ed}(t) = X_e(0) + (X_e(\infty) - X_e(0))\; g(t) \tag{40}$$

where $X_e(0)$ and $X_e(\infty)$ are the initial and final values of the tip position vector, and $g(.)$ is a cycloidal function of the form

$$g(v) = \begin{cases} \dfrac{v}{\tau} - \dfrac{1}{2\pi}\sin\dfrac{2\pi v}{\tau}, & 0 \le v \le \tau \\ 1, & v \ge \tau \end{cases} \tag{41}$$

where τ is the time required to move the tip from its initial position $X_e(0)$ its final position $X_e(\infty)$, and is selected as $\tau = 2s$. Note that (40)-(41) describe the equation of a straight line in the Cartesian

coordinates. The starting joint angle vector is selected as $\Theta(0) = (-45,\ -20,\ -5,\ 0,\ 50)^T$ degrees. These correspond to the tip Cartesian position $X_e(0) = (340,\ -130,\ 810)^T\ mm$. The final position of the tip is specified as $X_e(\infty) = (500,\ 500,\ 500)^T\ mm$, and the Cartesian distance to be traveled is $D = \|X_e(\infty) - X_e(0)\| = 720\ mm$.

4.1.1 Case A

In this case study, combined augmentation and optimization are utilized for the resolution of the redundancy. In addition to the tip trajectory tracking, it is desired to decrease the elbow height linearly from the starting value $h(0) \equiv X_a(0) = 150\ mm$ to reach a value of $h(\infty) = X_a(\infty) = 0$ at time $t = 1\,s$. This height is to be maintained to avoid collision with an obstacle while the tip continues its motion to get to the goal position at time $t = \tau = 2\,s$. The desired elbow height trajectory is

$$X_{ad}(t) = X_a(0) + (X_a(\infty) - (X_a(0))\ g(2t) \tag{42}$$

where $g(.)$ is given in (41). The desired posture vector is obtained by augmenting the tip position vector and the elbow height as

$$X_d(t) = \begin{pmatrix} X_{ed}(t) \\ X_{ad}(t) \end{pmatrix} = (x_{1d}(t),\ x_{2d}(t),\ x_{3d}(t),\ x_{4d}(t))^T \tag{43}$$

where the first three components describe the Cartesian coordinates of the tip position, and the last component, $x_{4d}(t)$, is the desired elbow height trajectory.

In addition to the above specifications and in order to avoid joint limits, it is desired to keep each joint angle θ_j, $j = 1, 2, \cdots, 5$, as close as possible to its center value $\theta_{jc} = \frac{\theta_{ju}+\theta_{jl}}{2}$, where θ_{ju} and θ_{jl} are the upper and lower limits of the joint angle θ_j. This is done by maximizing the function

$$\Psi(\Theta) = -\sum_{j=1}^{5} \left(\frac{\theta_j - \theta_{jc}}{\theta_{js}} \right)^6 \tag{44}$$

where $\theta_{js} = \frac{\theta_{ju}-\theta_{jl}}{2}$ is the span. The set of pairs $(\theta_{il}, \theta_{iu})$ for joints 1 to 5 of the PUMA are, respectively, $(-160, 160)$, $(-223, 43)$, $(-48, 236)$, $(-110, 170)$ and $(-100, 100)$ degrees.

The control algorithm (32), (33) and (37)-(39) are applied in which the gain matrix $K(t)$ is adapted with $K_P = K_0 = 0$, $\alpha = 10^9$ and $\sigma = 0.7$, so that only integral adaptation with a leakage term is employed. Note that the large value of α is due to the fact that the errors E_i and the sample time T_c in (32) are measured in meters and seconds, respectively. [1]. The values of $\beta_0 = 0.007$ and $w_0 = 0.015$ are used in (38). The pseudoinverse Jacobian matrix is computed every $T_J = 100\ ms$. This means that for every 50 control cycles only one pseudoinverse Jacobian is computed. In addition, to account for the delay in computation, a one sample delay of T_J is introduced in the calculations of the Jacobian pseudoinverse matrix, as discussed before. Thus the

[1] Although the results are given in millimeter, milliseconds and degrees, the actual computations are performed with units in meters, seconds and radians

pseudoinverse being used for computing of the joint angles is out of date by as much as 200 ms, which can produce large mismatch.

The arm motion is shown in Figure 3 . The actual tip position trajectories ($x_1(t)$, $x_2(t)$, $x_3(t)$) and the elbow height trajectory $x_4(t)$ closely follow the desired trajectories ($x_{1d}(t)$, $x_{2d}(t)$, $x_{3d}(t)$) and $x_{4d}(t)$, respectively, as seen in Figure 4. The errors $e_1(t) = x_{1d}(t) - x_1(t)$, $e_2(t) = x_{2d}(t) - x_2(t)$, $e_3(t) = x_{3d}(t) - x_3(t)$ and $e_4(t) = x_{4d}(t) - x_4(t)$ are plotted in Figure 5 and indicate maximum errors of 1.0 mm, 0.7 mm, 0.6 mm along the three axis, and 1.05 mm in the elbow height. These maximum errors are very small compared to the traveled distances shown in Figure 4. The average errors are less than 0.3 mm. It must be noted that the above low tracking errors have been obtained despite relatively fast trajectories and infrequent Jacobian matrix computation. The joint angle trajectories to achieve the above motion is shown in Figure 6. The joint angles have been kept as close as possible to their center values through the optimization of the performance criterion. It is interesting to note from Figures 3 and 6 that the arm continues to change posture to further optimize the performance criterion even at the completion of the tip trajectory. This is done while keeping both the tip at the position $X_e(\infty) = (500,\ 500,\ 500)^T$ mm and the elbow at the height of $X_a(\infty) = 0$. The optimization can be speeded up and distributed more evenly throughout the arm motion by increasing the value of γ in (15).

Suppose now that to further reduce the computations, the Jacobian pseudoinverse matrix is initialized correctly, but is not updated at all, that is $T_J = \infty$. The norm of the mismatch matrix, $\|H\|$, is plotted in Figure 7 and indicates a maximum mismatch of $\eta = max\|H\| \approx 2.8$ which exceeds the conservative limit of 1 used in the proof of stability. Despite this, the error responses plotted in Figure 8 show a maximum error of only 1.3 mm, 0.8 mm and 1.3 mm in the three coordinates of the tip position and 0.3 mm in the elbow height. These error are only slightly higher than those obtained with 100 ms updating. It is seen that despite suspension of the Jacobian computations, which has resulted constant Jacobian matrix through the entire motion, the scheme has tracked the desired trajectories accurately. This has been achieved mainly due to the adaptive control scheme.

Finally, let us quantify the computational savings as a result of infrequent updating. Let the CPU time required to compute the joint trajectory vector Θ for one control cycle with Jacobian updating times $T_J = T_c = 2\,ms$, $T_J = 100\,ms$ and $T_J = \infty$ be denoted by t_2, t_{100} and t_∞, respectively. Using a Sun computer it is found that for the present example, $t_2 = 6.5\,ms$, $t_{100} = 1.06\,ms$ and $t_\infty \doteq 1.00\,ms$. These results reflect the computational intensity involved in the Jacobian pseudoinverse calculations despite using an efficient algorithm [23]. The results also indicate considerable CPU time savings due to infrequent updating of $T_J = 100\,ms$. Although the adaptive scheme can tolerate no pseudoinverse updating, only little extra savings is obtained using $T_J = \infty$.

4.1.2 Case B

The purpose of this case study is to show how both extra degrees of freedom can be used for augmentation, and to further demonstrate the robustness of the algorithm.

Consider the manipulator where the tip and the elbow are required to track the trajectories as in Case A. In addition, these tasks are to be performed while freezing one wrist joint angle to a value of $\theta_4 = 45^0$ to represent the case of a joint motor failure. The additional task vector is now the 2×1 vector $X_a = \begin{pmatrix} x_4 \\ x_5 \end{pmatrix}$ where $x_5 = \theta_4$, and the posture vector is $X = (X_e \quad X_a)^T = (x_1, x_2, \cdots, x_5)^T$. All parameters are left at their values as in Case A. The error responses are shown in Figures 9 and 10 and indicate low tracking errors in all components of the posture vector. The maximum Cartesian error occurs in x_1 with a value of $e_1 = 0.9\ mm$, and the average error is less than $0.4\ mm$. The maximum wrist angle deviation from its frozen value is 0.6 degree. The joint angles trajectories to achieve the above tasks are shown in Figure 11, and reach their final values after about $2\,s$ when the desired trajectories are their steady-state values. Comparing Figures 11 and 6, we observe that the joint angles in Figure 11 are farther away from their center values because no optimization is performed in Case B. The arm motion is depicted in Figure 12 and further illustrates this point.

If the inverse Jacobian matrix is not updated, the maximum Cartesian error is found to be $1.05\ mm$, and the maximum wrist angle error is 0.3 degree, which are about the same values with $T_J = 100\ ms$. The CPU times for one control cycle computation are $t_2 = 6.0\,ms$, $t_{100} = 0.80\,ms$ and $t_\infty = 0.62\,ms$ for Jacobian update times of $2\,ms$, $100\,ms$ and ∞, respectively. These values are close to those obtained in Case A, and further illustrate the significant CPU time savings as a result of infrequent updating.

4.2 Example 2

The purpose of this example is to demonstrate the capability of the adaptive scheme to perform both position and orientation trajectory tracking in the standard PUMA 650 robot in the difficult situation where the manipulator is required to perform a complex motion that passes through a robot singularity repeatedly. Orientation trajectory tracking is generally hard to achieve due to the presence of many singularities associated with the wrist. The end-effector vector is denoted by $X = (x_1, x_2, \cdots, x_6)^T$, where x_1, x_2 and x_3 are the position components, and x_4, x_5 and x_6 are the orientation components. Here, all degrees of freedom are used for the primary tasks. The end-effector coordinate vector X is related to the angles of waist θ_1, shoulder θ_2, elbow θ_3, and the three wrist angles θ_4, θ_5 and θ_6 through the forward kinematic equations given in the Appendix. The orientation is described by the equivalent axis representation [24].

The desired end-effector trajectory is selected as shown in Figure 13. The PUMA manipulator starts at $\theta_1 = \theta_2 = \theta_3 = \theta_4 = \theta_6 = 0$, and $\theta_5 = 30^o$ with the corresponding end-effector position $(x_1\ ,\ x_2\ ,\ x_3)^T = (440,\ 149,\ 481)^T$ mm, and the orientation $(x_4\ ,x_5\ ,x_6)^T = (0,\ 30,\ 0)^T$ degrees. The end-effector takes a partial spiral path until it reaches a tilted circular path of a specified radius.

It then continues its motion on this circular path repeatedly. The desired end-effector orientation is such that it holds an object upward (e.g. holds a glass of water without spilling the water) while the tilted path is traversed. In order to specify this tilted spiral-circular motion, we defined the rotation matrices:

$$R_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi_1 & -\sin\varphi_1 \\ 0 & \sin\varphi_1 & \cos\varphi_1 \end{pmatrix} \; ; \; R_2 = \begin{pmatrix} \cos\varphi_2 & 0 & \sin\varphi_2 \\ 0 & 1 & 0 \\ -\sin\varphi_2 & 0 & \cos\varphi_2 \end{pmatrix} \tag{45}$$

where φ_1 and φ_2 are angles of rotation about x_1 and x_2 axis, respectively. Now we form the position and orientation vectors p and and q as

$$p(t) = R_2 R_1 \begin{pmatrix} r\sin(\frac{2\pi t}{\tau}d) \\ r\cos(\frac{2\pi t}{\tau}d) \\ 0 \end{pmatrix} \; ; \; q = \begin{pmatrix} x_4(\infty) - x_4(0) \\ x_5(\infty) - x_5(0) \\ x_6(\infty) - x_6(0) \end{pmatrix} \tag{46}$$

where $d = \pm 1$ specifies the direction of rotation. Note that $p(t)$ describes the equation of a circle that is rotated about axis x_1 and x_2. The desired trajectory vector is now

$$X_d(t) = X(0) + \begin{pmatrix} p \\ q \end{pmatrix} g(4t) \tag{47}$$

where g(.) is given in (41), and has the effect of smoothly taking the end-effector from the starting position and orientation $X(0)$ into a spiral path and finally to a circular path with the desired tilt. The parameters in (45)-(46) are selected as $\varphi_1 = 35$ degrees $\varphi_2 = 20$ degrees, $r = 300\,mm$, $\tau = 5\,s$ and $d = -1$.

The parameters are $K_0 = K_P = 0$, $\alpha = 10^7$, $\sigma = 1.5$, $\beta_0 = 0.01$, $w_0 = 0.025$ and the Jacobian matrix update time is $T_J = 100\ ms$. The determinant of the Jacobian matrix is shown in Figure 14, and clearly changes sign, showing that the manipulator passes through a singularity repeatedly. The position and orientation errors are given in Figures (15) and (16) . The maximum position and orientation errors are about 3.5 mm and 0.7 degree, and occur along x_1 and x_4, respectively, and occur at the singularity point. The joint velocities are plotted in Figure 17 and show a maximum value of less than 120 degrees/s as the arm goes through singularities. It is seen that despite infrequent Jacobian updating and the complex manipulator motion passing through singularities, the adaptive kinematic control scheme has produced low errors and velocities. The CPU time savings due to infrequent updating is again significant in this case with $t_2 = 7\,ms$ and $t_{100} = 1.0\,ms$.

It must be emphasized that the above two examples demonstrate typical performance of the proposed adaptive scheme. Similar results were obtained for a wide range of desired trajectories with different velocities and with very infrequent Jacobian pseudoinverse computations. Higher tracking accuracies are achieved by more elaborate tuning of the controller parameters K_P, K_0, α and σ.

Finally, the scheme was implemented on both the standard and modified PUMA 562. The control software was written in the C language on a 486 PC. A serial link was established for the communication between the higher level adaptive controller residing on the PC and the PUMA joint level controller. Several examples similar to the above were successfully implemented.

5 Conclusions

A method is proposed for redundancy resolution that generalizes Jacobian pseudoinverse and augmentation approaches. The method is flexible and allows the user to easily define tasks involving trajectory tracking and performance optimization. An adaptive algorithm for solving the inverse kinematic problem of redundant robot has been presented which uses a feedback loop with an adaptive controller to generate the joint angle trajectories for achieving desired end-effector trajectories, as well as other desired posture trajectories defined by the user.

It is shown that the errors in posture trajectories are globally ultimately bounded for different velocities and bounded uncertainties in the estimation of the Jacobian pseudoinverse matrix. This robustness allows very infrequent pseudoinverse computations and make the scheme computationally fast and suitable for real-time implementation. A further feature of the scheme is its tolerance to manipulator singularities and allows the robot to go close to or even pass through singularities while maintaining acceptable joint velocities and low tracking errors.

The scheme has been applied to a practical redundant robot for achieving both trajectory tracking and optimization and the results show the effectiveness of the scheme under a variety of conditions.

6 References

[1] D.E. Whitney, "Resolved motion rate control of manipulators and human prosthesis," IEEE Trans. Man Machine Sys. vol. MMS-10, No. 2, 47-53, 1969.

[2] A. Liegeois, "Automatic supervisory control of configuration and behavior of multibody mechanisms," IEEE Trans. System, Man, and Cybernetics, vol. SMC-7, No. 12, pp. 868-871, 1977.

[3] A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically time-varying environments," Int. J. Robotics Res., vol. 4, No. 3, pp. 109-117, 1985.

[4] J. M. Hollerback and K. C. Suh, "Redundancy resolution of manipulators through torque optimization,", Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1016-1021, 1885.

[5] N. Nakamura and H. Hanafusa, "Task priority based redundancy control of robot manipulators," Proc. 2nd Int. Symp. on Robotics Research, Kyoto, Japan, 1984.

[6] N. Nakamura, Advanced Robotics - Redundancy and Optimization, Chapt. 4, Addison-Wesley Publishing Co., 1991

[7] C. A. Klein and C. H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," IEEE Trans, System, Man and Cybernetics, vol. SMC-13, No. 3, pp. 245-250, 1983.

[8] H. Hanafusa, T. Yoshikawa and Y. Nakamura, "Analysis and control of articulated robot arm with redundancy," Proc. 8th IFAC Triennial World Congress, pp. 1927-1932, Kyoto, Japan, 1981.

[9] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," in Robotics

and Research, eds. M. Brady and R. Paul, pp. 735-747, MIT Press, 1984.

[10] J. Bailieul, J. Hollerbach and R. Brockett, "Programming and control of kinematically redundant manipulators," Proc. 23rd IEEE Conf. on Decision and Control, pp. 768-774, Las Vagas, NV, 1984.

[11] J. Bailieul, "Kinematic programming alternatives for redundant manipulators," Proc. IEEE Int. Conf. on Robotics and Automation, pp. 722-728, St. Louis, 1985.

[12] J. Bailieul, "Avoiding obstacles and resolving kinematic redundancy," Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1698-1704, San Francisco, CA, 1986.

[13] H. Seraji, ""Configuration control of redundant manipulators: Theory and implementation," IEEE Trans. on Robotics and Automation, vol. RA-5, pp. 472-490, 1989.

[14] H. Seraji and R. Colbaugh, "Improved configuration control for redundant robots," J. Robotics Systems, Vol. 7, No. 6, pp. 897-928, 1990.

[15] N. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," ASME J. Dyn. Syst., Meas. and Control, vol. 108, pp. 163-171, 1986.

[16] W. A. Wolovich and H. Elliot, "A Computational Technique for Inverse Kinematics", Proc. 23 IEEE Conf. on Decision and Control, Las Vegas, NV., 1984, pp. 1359-1363

[17] R. J. Vaccaro, and S. D. Hill, "A Joint-Space Command Generator for Cartesian Control of Robotic Manipulators",IEEE Journal of Robotics and Automation, Vol. 4, No. 1, pp. 70-76, 1988.

[18] L. Sciavico and B. Siciliano, "A dynamic solution to the inverse kinematic problem for redundant manipulators," Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1081-1087, Raliegh, NC, 1987.

[19] T. C. Hsia and Z. Y. Guo, "New inverse kinematic algorithms for redundant robots," J. Robotic Systems, vol. 8, No. 1, pp. 117-132, 1991.

[20] R. Colbaugh, K. Glass and H. Seraji, "An Adaptive Inverse Kinematic Algorithm for Robot Manipulators," Int. J. Modelling and Simulation, vol. 11, No. 2, pp. 33-38, 1991.

[21] P. A. Ioannou and P. V. Kokotovic, Adaptive Systems with Reduced Models, Springer-Verlag, 1983.

[22] D. E. Whitney, "The mathematics of coordinated control of prosthese and manipulators," J. Dyn. Sys., Meas., Control, vol. 94, No. 4, oo. 303-309, 1972.

[23] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, Numerical Recipes in C, Press Syndicate of the University of Cambridge, New York, NY, pp. 39-47, 1990

[24] J. J. Craig, Introduction to Robotics - Mechanics and Control, Addison Wesley, Reading, MA, 1986.

[25] M. I. Vuskovic, Advanced Robotics, Lecture Notes, Chapt. 3, Department of Mathematical Sciences, San Diego State University, CA, 1988.

7 Appendix - Forward Kinematic Equations

In this appendix we present the forward kinematic equations of the five DOF redundant manipulator used in Example 1, and the 6 DOF PUMA 562 used in Example 2.

Consider the forward kinematics of the modified PUMA 650 manipulator shown in Figure (reffinarmpic.ps). Let θ_1, θ_2 and θ_3 be the waist, shoulder and elbow joint angles, respectively, and use the standard notations $c_1 = cos\theta_1$, $s_1 = sin\theta_1$, $c_{12} = cos(\theta_1 + \theta_2)$, etc. The shoulder position vector p_{sh} is

$$p_{sh} = \begin{pmatrix} -d_1 s_1 \\ d_1 c_1 \\ 0 \end{pmatrix} \tag{48}$$

where $d_1 = 149.1\ mm$ is the base to shoulder length . The unit vector $a_2 = \frac{p_{sh}}{\|p_{sh}\|}$ defines the reference axis for θ_2. The elbow position vector is

$$p_{el} = p_{sh} + \begin{pmatrix} l_2 c_1 c_2 - l_3 c_{23} c_1 \\ l_2 s_1 c_2 \\ -l_2 s_2 + l_3 s_{23} \end{pmatrix} \tag{49}$$

where $l_2 = 432\ mm$ is shoulder joint to elbow joint distance and $l_3 = 20.3\ mm$ is the elbow offset. The elbow height is $h = -l_2 s_2$. The wrist position is

$$p_{wr} = p_{el} + v \equiv p_{el} + \begin{pmatrix} d_4 c_1 s_{23} \\ d_4 s_1 s_{23} \\ d_4 c_{23} \end{pmatrix} \tag{50}$$

where $d_4 = 432\ mm$ is elbow to wrist length. The unit vector $a_4 = \frac{v}{\|v\|}$ defines the reference axis for θ_4. To find a_5, the unit vector for θ_5, we note that due to the PUMA 562 geometry, a_5 is parallel to a_2 when the manipulator is at "home" position and θ_5 is zero. Consequently, the unit vector a_5 is obtained as $a_5 = rot(\theta_4, a_4)\, a_2$ where $rot(\theta_4, a_4)$ is the matrix representing the rotation about the vector a_4 by θ_4, and is obtained from

$$rot(\theta_4, a_4) = c_4 I + (1 - c_4) a_4 a_4^T + s_4 S(a_4) \tag{51}$$

where $S(.)$ is the skew symmetric operator defined so that for two vectors a and b, $a \times b = S(a)b$. Since a_6, the reference axis for joint θ_6, is parallel with a_4 when $\theta_5 = 0$, we have $a_6 = rot(\theta_5, a_5)a_4$ which rotates a_4 about a_5 by θ_5. The position of the end-effector mounting plate is

$$p_e = l_5 a_6 + p_{wr} \tag{52}$$

where $l_5 = 57.2\,mm$ is the wrist to plate length. When the extra link is mounted on the end-effector plate, the tip position of this link is

$$X_{tip} = (l_5 + l_6) a_6 + p_{wr} \tag{53}$$

where $l_6 = 250\,mm$ is the length of the extra link.

The orientation of the plate is determined using the equivalent axis representation [24]. This requires computing the rotation matrix

$$R = rot(\theta_6, a_6)\, rot(\theta_5, a_5)\, rot(\theta_4, a_4)\, rot(\theta_3 + \theta_2, a_2)\, rot(\theta_1, a_1) \tag{54}$$

Note that $a_2 = a_3$ since the axis of rotation of joints 2 and 3 are parallel. Given the rotation matrix R, the angle of rotation ϑ and the axis of rotation k are obtained from [24]

$$\vartheta = \cos^{-1} \frac{1}{2}(r_{11} + r_{22} + r_{33} - 1) \tag{55}$$

where $r_{11}, \cdots, r_{33}$ are the elements of the rotation matrix R. The end-effector orientation is

$$\phi_e = \omega \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}, \quad \vartheta \leq \pi - \epsilon \tag{56}$$

where ϵ is a small number and

$$\omega = \begin{cases} \dfrac{\vartheta}{2 \sin \vartheta}, & \epsilon < \vartheta < \pi - \epsilon \\ \dfrac{1}{2}, & \vartheta \leq \epsilon \end{cases} \tag{57}$$

In order to determine ϕ_e for $\vartheta > \pi - \epsilon$, let $u_1 = \sqrt{\frac{r_{11}+1}{2}}$, $u_1 = \sqrt{\frac{r_{22}+1}{2}}$, and $u_1 = \sqrt{\frac{r_{33}+1}{2}}$. Then it can be shown that [26] $\phi_e = \vartheta \begin{pmatrix} u_1 \\ \frac{r_{12}}{2u_1} \\ \frac{r_{13}}{2u_1} \end{pmatrix}$ for $\vartheta \geq \pi - \epsilon$ and $|u_1| > \epsilon$; $\phi_e = \vartheta \begin{pmatrix} \frac{r_{21}}{2u_2} \\ u_2 \\ \frac{r_{23}}{2u_2} \end{pmatrix}$ for $\vartheta \geq \pi - \epsilon$, $|u_1| < \epsilon$ and $|u_2| > \epsilon$; and $\phi_e = \vartheta \begin{pmatrix} \frac{r_{31}}{2u_3} \\ \frac{r_{32}}{2u_3} \\ u_3 \end{pmatrix}$ for all other values of ϑ. Finally, the end-effector position and orientation vector is $X_e = \begin{pmatrix} p_e \\ \phi_e \end{pmatrix}$.

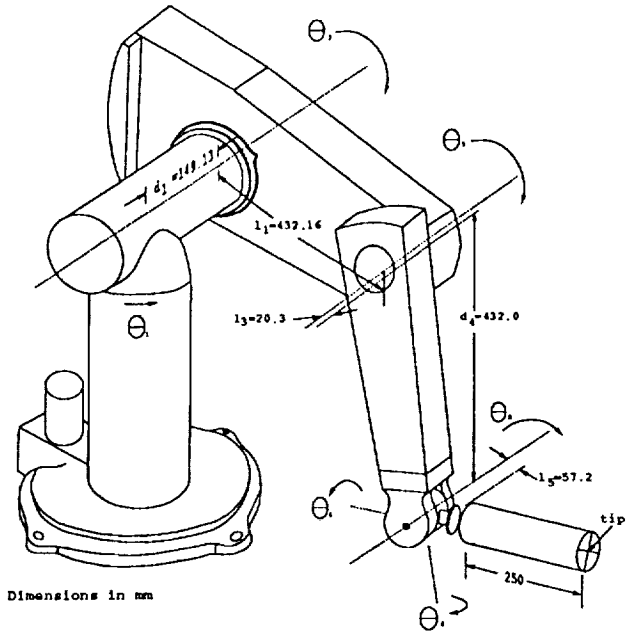

Figure 2: The modified PUMA 562 Manipulator.

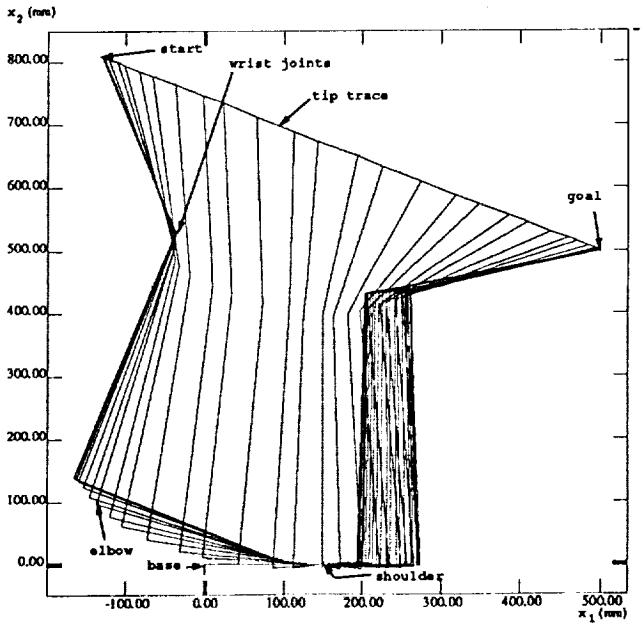

Figure 3: Trace of arm motion, Example 1 - Case A.

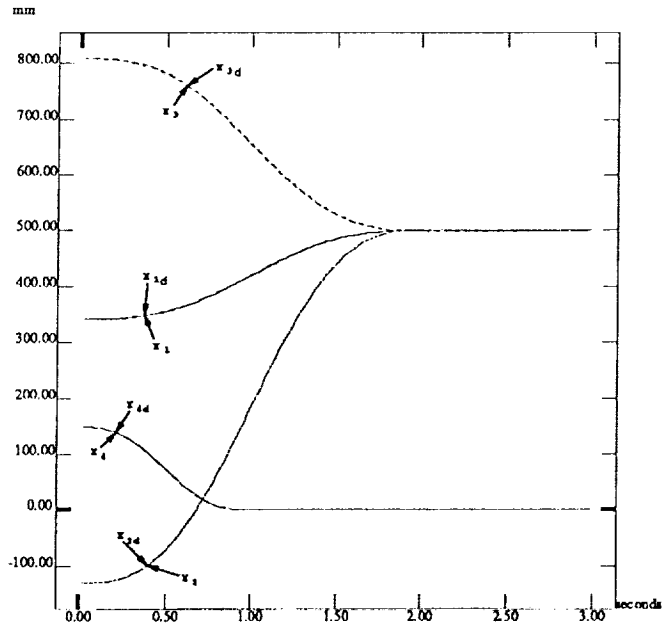

Figure 4: Desired and actual Cartesian trajectories, Example 1 - Case A.

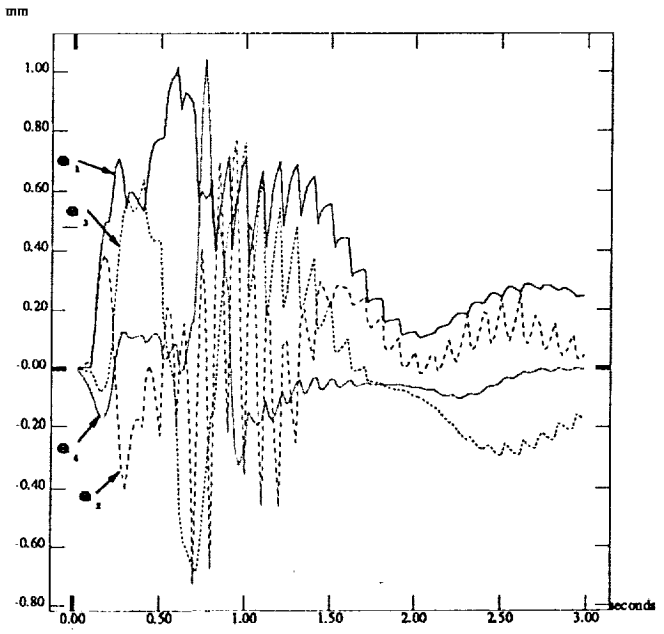

Figure 5: Errors responses, Example 1 - Case A.

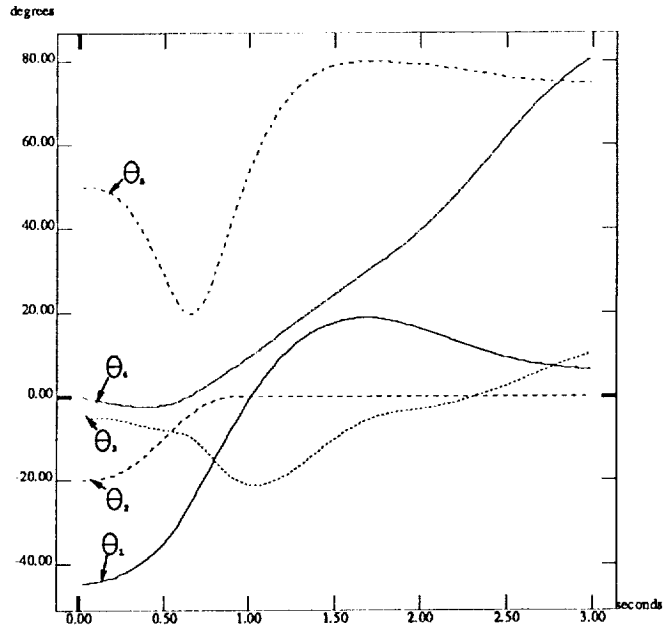

Figure 6: Joint angle trajectories, Example 1 - Case A.

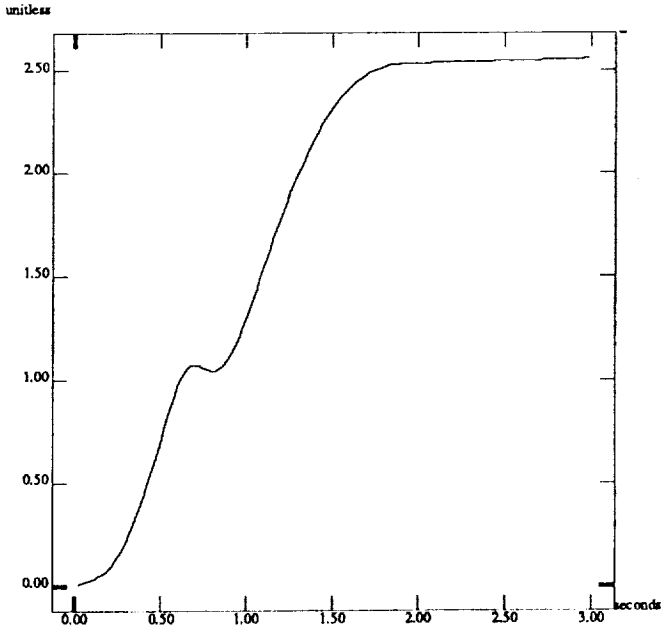

Figure 7: Norm of the mismatch matrix $\|H\|$ with no Jacobian updating, Example 1 - Case A.

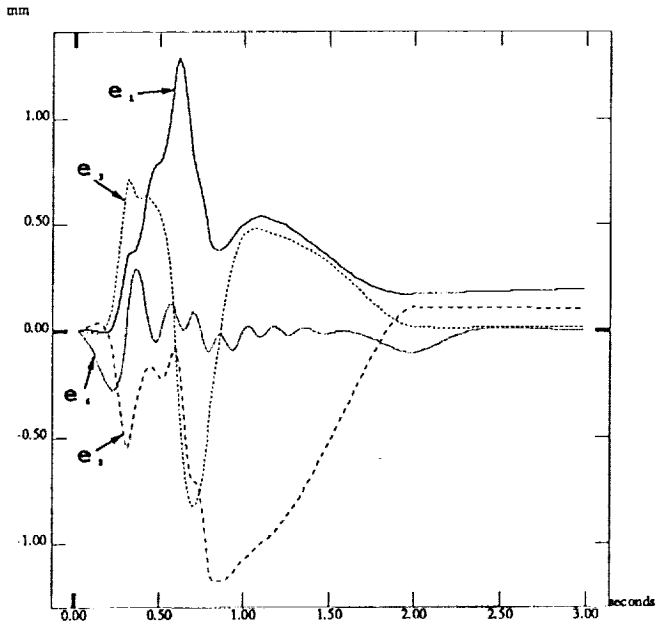

Figure 8: Error responses without Jacobian updating, Example 1 - Case A.

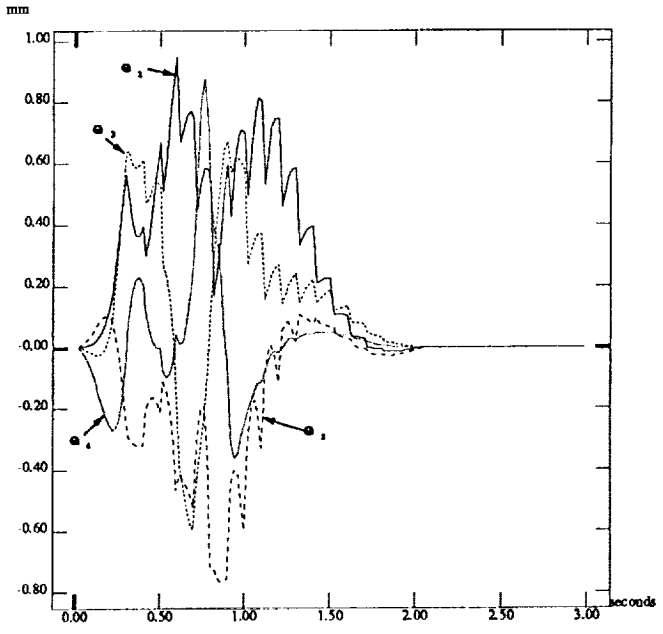

Figure 9: Error responses, Example 1 - Case B.

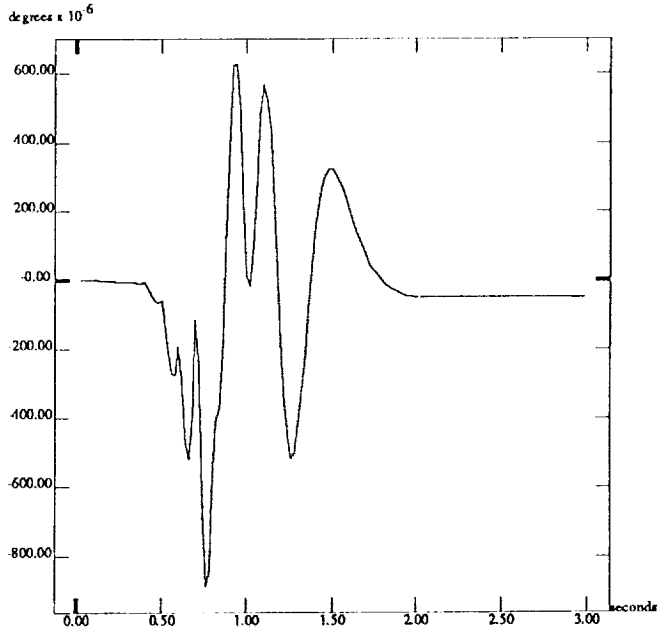

Figure 10: wrist angle error, Example 1 - Case B.

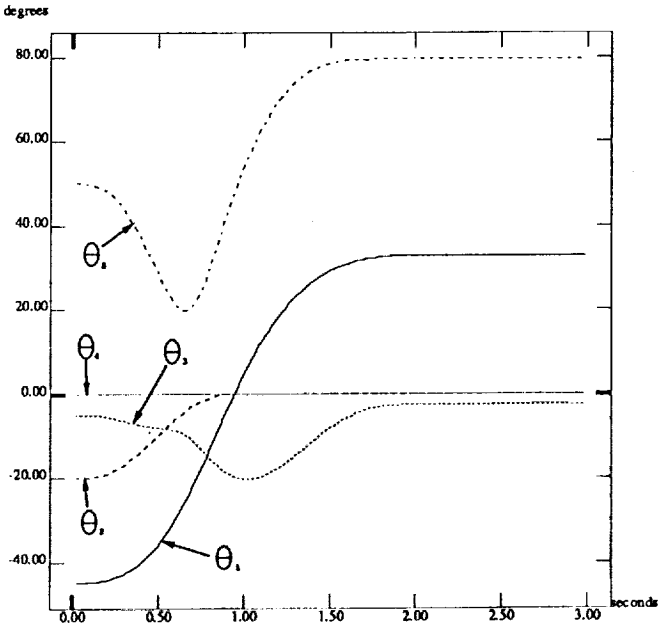

Figure 11: Joint angle trajectories, Example 1 - Case B.

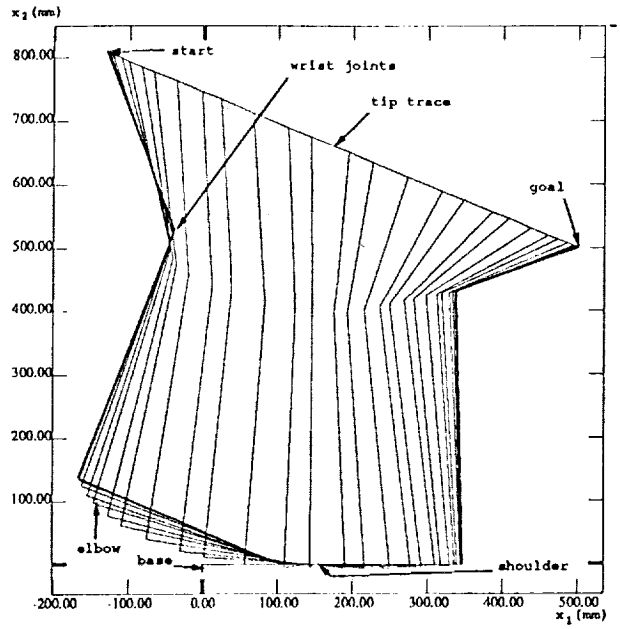

Figure 12: Trace of the arm motion, Example 1 - Case B.

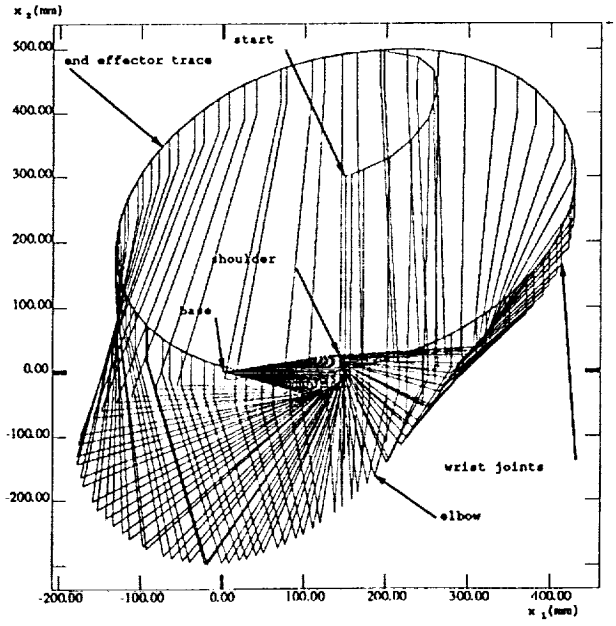

Figure 13: Arm motion, Example 2.

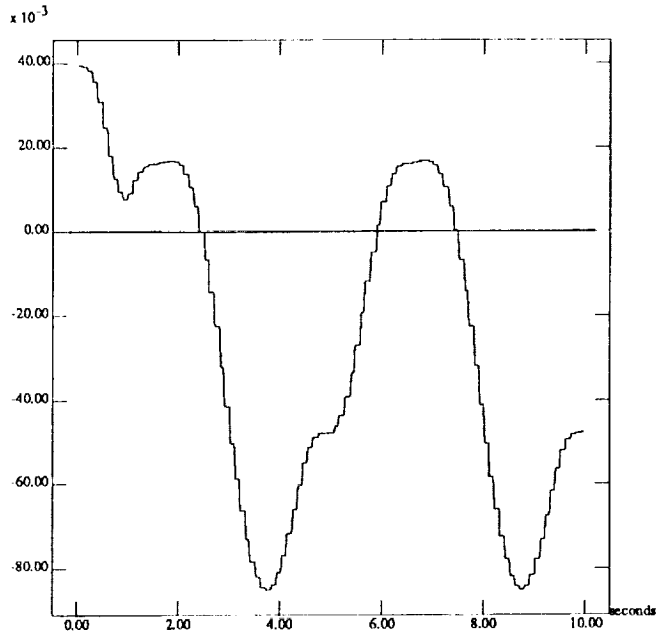

Figure 14: Determinant of the Jacobian matrix, Example 2.

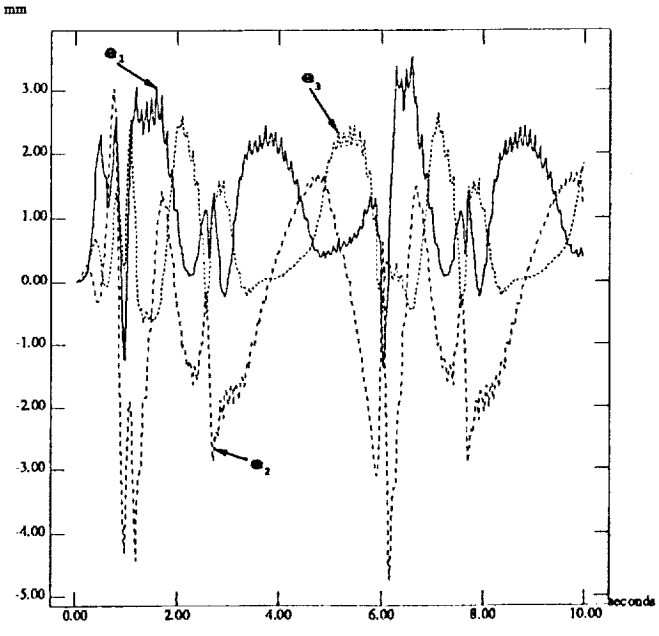

Figure 15: End-effector position error trajectories, Example 2.

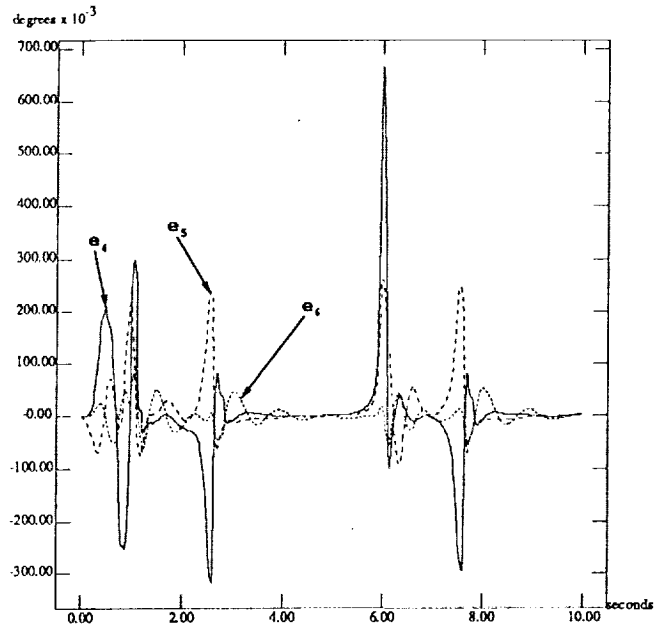

Figure 16: End-effector orientation error trajectories, Example 2.

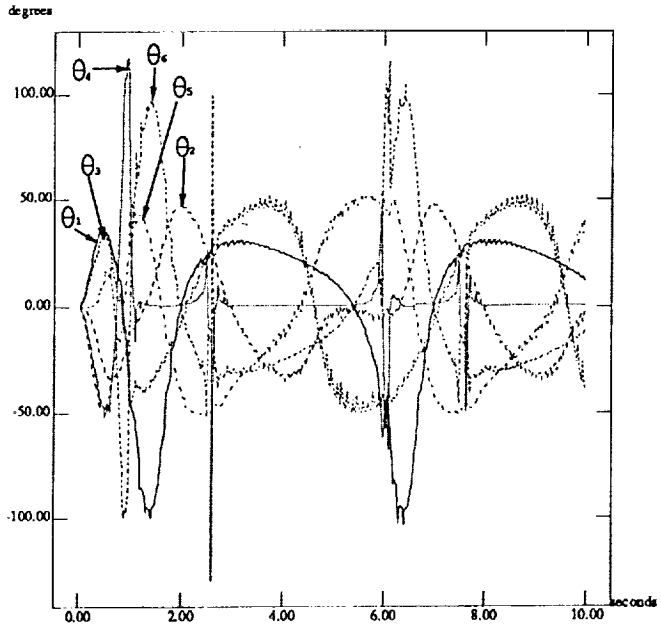

Figure 17: Joint velocity trajectories, Example 2.